Additional Labscript for:

CMSIS and TIMER on the ARM Cortex 4 Microcontroller

Author: Ibrahim Ozturk.

Course: Design and Construction (Module ELE00003H)

Introduction

Instead of traditional way of code implementation, you can also try The Cortex Microcontroller Software Interface Standard (CMSIS) which will give you amazing opportunity to port and reuse your software in various projects. Furthermore, CMSIS is really user friendly, easy to catch and track when you are coding.

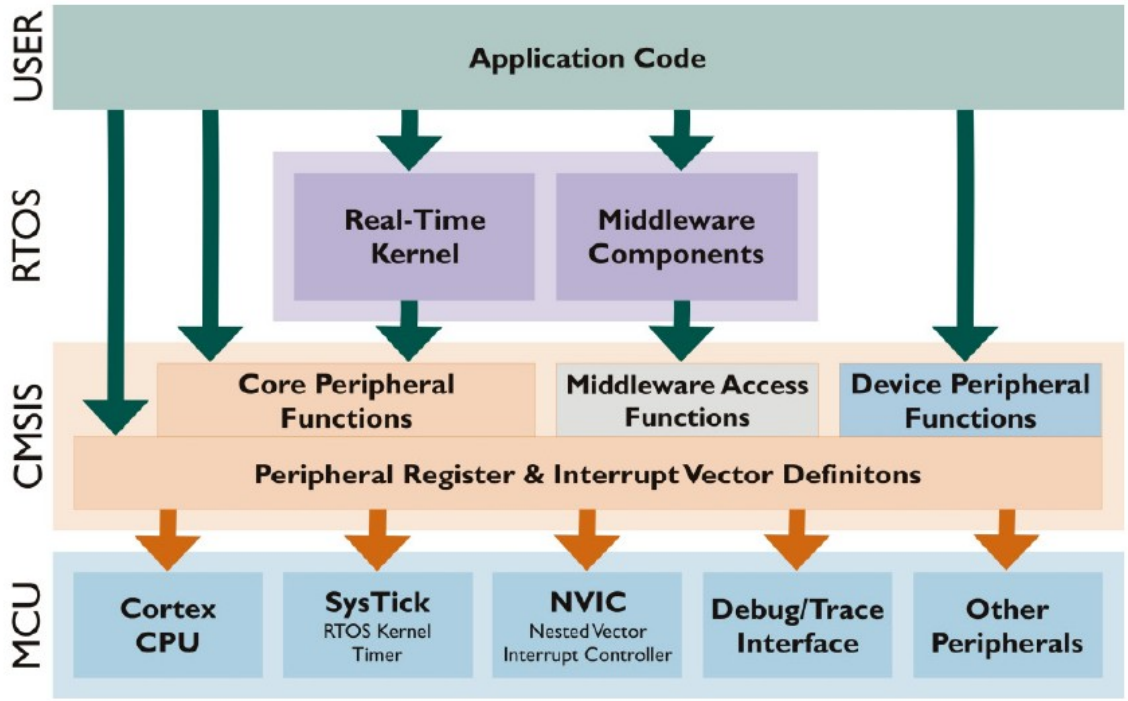

Figure 1: CMSIS Structure functional flow [1]

Part 1: Comparison

In order to explain difference between CMSIS and traditional way in real environment, I am going to look at a simple example on STM32F4Discovery and York I/O Board. The example programs will configure clock and GPIO initialization in both ways by compiling separate project files for the Keil µVision IDE.

Those two GPIO Pins initialization implementations on Figure 2 and Figure 3 are almost identical in terms of what they are configuring General Purpose Input-Output pins.

```
1 void LED_Init (void)
2 {
3   RCC->AHB1ENR  |= ((1UL <<  3) );        /* Enable GPIOD clock             */
4
5   GPIOD->MODER    &= ~((3UL << 2*12) |
6                        (3UL << 2*13) |
```

```
                (3UL << 2*14) |
                (3UL << 2*15)  );   /* PD.12..15 is output              */
  GPIOD->MODER    |=  ((1UL << 2*12) |
                (1UL << 2*13) |
                (1UL << 2*14) |
                (1UL << 2*15)  );
  GPIOD->OTYPER   &= ~((1UL <<   12) |
                (1UL <<   13) |
                (1UL <<   14) |
                (1UL <<   15)  );   /* PD.12..15 is output Push-Pull     */
  GPIOD->OSPEEDR  &= ~((3UL << 2*12) |
                (3UL << 2*13) |
                (3UL << 2*14) |
                (3UL << 2*15)  );   /* PD.12..15 is 50MHz Fast Speed     */
  GPIOD->OSPEEDR  |=  ((2UL << 2*12) |
                (2UL << 2*13) |
                (2UL << 2*14) |
                (2UL << 2*15)  );
  GPIOD->PUPDR    &= ~((3UL << 2*12) |
                (3UL << 2*13) |
                (3UL << 2*14) |
                (3UL << 2*15)  );   /* PD.12..15 is Pull up             */
  GPIOD->PUPDR    |=  ((1UL << 2*12) |
                (1UL << 2*13) |
                (1UL << 2*14) |
                (1UL << 2*15)  );
}
```

Figure 2: GPIO Pins initialization by traditional way

```
void Led_Init(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

  /* Configure PD12, PD13, PD14 and PD15 in output pushpull mode */
  //Main 4 LEDs on STM32F4 and I/O Board
  GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_12 | GPIO_Pin_13| GPIO_Pin_14| GPIO_Pin_15;
  GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
  GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

Figure 3: GPIO Pins initialization by CMSIS

After having a quick look to both styles, you will see how CMSIS implementation as in Figure 3 is easy and user friendly compared with traditional way in Figure 2. It is up to you which one you want to prefer in your further implementations.

Part 2: Timer

Instead of using a busy-wait approach to timing, it is more efficient to make this interrupt driven which we will show the implementation in this tutorial.

```
void TIMER3_Config (void)
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    /* TIM3 Clock Enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    /* Enable TIM3 Global Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

```

```
    NVIC_Init(&NVIC_InitStructure);

    /* Time Base Configuration */
    /* PrescalerValue = (uint16_t) ((SystemCoreClock / 2) / 28000000) -1;*/

    TIM_TimeBaseStructure.TIM_Period =  100 - 1; // 1 MHz down to 10 KHz (0.1 ms)
    TIM_TimeBaseStructure.TIM_Prescaler =  84 - 1; // DownTo 1MHz (adjust per your clock)
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);

    /* TIM Interrupts ENABLE*/
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);

    /*Enable TIM3 Counter */
    TIM_Cmd(TIM3, ENABLE);
}
```

Figure 4: TIMER Initialize with Interrupt

```
int counter;
void TIMER3_IRQHandler (void)
{
    counter++;
    if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
    {
      TIM_ClearITPendingBit(TIM3, TIM_IT_Update | TIM_IT_CC1);
      /* procedure for interrupt at phase change*/
      //If you want to toggle LED connected PD12, uncomment this
      //GPIO_ToggleBits(GPIOD, GPIO_Pin_12);
    }
    else
    {
      TIM_ClearITPendingBit(TIM3, TIM_IT_CC1);
      /* procedure for interrupt delivering data output */
    }
}
```

Figure 5: TIMER Interrupt Handler

```
int main (void)
{
   TIMER3_Config();
   while (1)
   {
    /* Do whatever you want to do */
   }
}
```

Figure 6: main.c design

Part 3: Created Project for CMSIS

To run the given project based on CMSIS without problems, **your directory might look as below or you should configure your project based on your directory settings**. *Libraries* section in Figure 7 is coming from ST as "*STSW-STM32065STM32F4 DSP and standard peripherals library*" and you can download it from [this link](): http://www.st.com/web/en/catalog/tools/PF257901 . After downloading it, extract the zip file and put this *Libraries* folder under your project folder as it is shown in project tree. On the other hand, as the main part of the project you can download *Project* and *Utilities* from the [lecture website]() or from [here]().

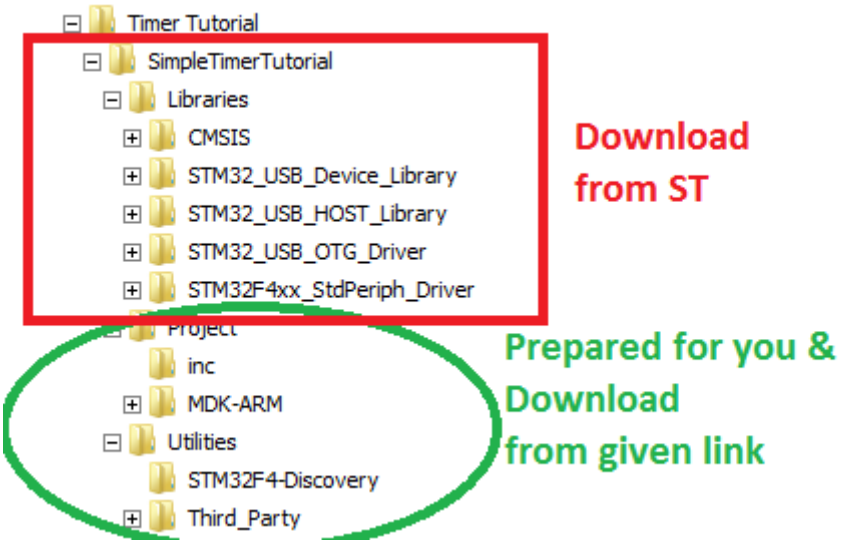

Figure 7: Project File Tree

As you can see on Figure 8, the project is compiled without any error and any warning.

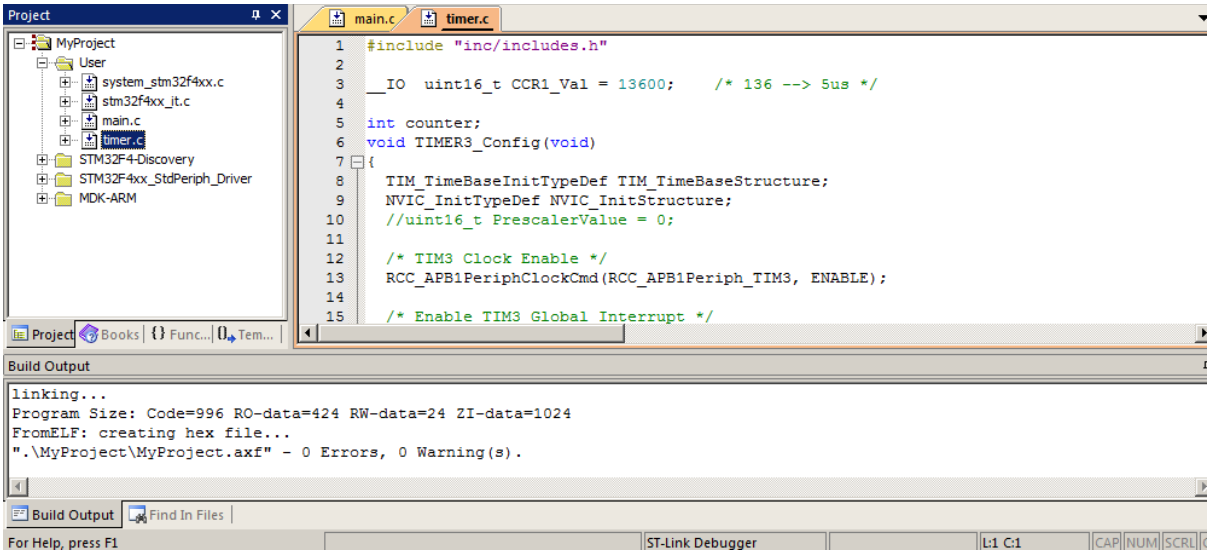

Figure 8: Compilation of the given project.

References

1. Getting started with CMSIS. (2009). Retrieved February 02, 2014, from https://www.doulos.com/knowhow/arm/CMSIS/CMSIS_Doulos_Tutorial.pdf
2. Projects on STM32F4Discovery Board. (2013). Retrieved February 02, 2014, from http://www.ozturkibrahim.com/projects/stm32fdiscovery/